REMARKS

This Amendment is in response to the Final Office Action dated June 3, 2005 ("FOA"). In the Office Action, claims 1-6 and 10 were rejected under 35 USC §102 and claims 7-9 were rejected under 35 USC §103. By this Amendment, claims 1, 2, 5 and 6 are amended. Currently pending claims 1-10 are believed allowable, with claims 1, 6 and 10 being independent claims.

CLAIM REJECTIONS UNDER 35 USC §102:

Claims 1-6 and 10 were rejected under 35 USC §102 as being anticipated by U.S. Patent No. 5,996,063 to Gaertner et al. ("Gaertner").

A *prima facie* case for obviousness can only be made if the combined reference documents teach or suggest all the claim limitations. MPEP 2143.

Claim 1:

Claim 1 is amended herein to recite, "bypassing a portion of the instruction pipeline for the current instruction if the no-dependency signal is active." Support for this claim limitation can be found at least at Fig. 6 and paragraph 62 of the present application. Claim 1, as amended, overcomes the 35 USC §102 rejection since Gaertner does not disclose bypassing a portion of the instruction pipeline for the current instruction if the no-dependency signal is active.

Gaertner is directed to allocating and renaming registers in a computing system that processes instructions out-of-order. Gaertner, col. 1, ln. 6-10. Specifically, an allocator is disclosed which is responsible for correctly addressing physical registers at the register renaming and allocation stage. Gaertner, col. 13, ln. 4-6 and Fig. 7. It is respectfully submitted that there is no teaching in Gaertner of bypassing a portion of an instruction pipeline for a current instruction if a no-dependency signal is active.

Claim 1 further recites, "for detection of a dependency, determining for each current instruction involved in a renaming process that a logic target address of one or more instructions is not the same as a logic source address of said current instruction, said one or more instructions being stored in a temporary buffer associated with a pipeline process downstream of the current instruction." In rejecting claim 1, the Office Action states that Gaertner teaches for detection of a dependency, determining for each current instruction involved in a renaming process that a logic target address of one or more instructions is not the same as a logic source address of said current instruction, said one or more instructions being stored in a temporary buffer associated with a pipeline process downstream of the current instruction. FOA, ¶ 6, citing Gaertner, col. 8, ln. 6-54 and Figs. 2-4. The Applicant respectfully disagrees with such an interpretation of Gaertner.

Fig. 2 shows a program example executed out-of-order. It does not illustrate for detection of a dependency, determining for each current instruction involved in a renaming process that a logic target address of one or more instructions is not the same as a logic source address of said current instruction, said one or more instructions being stored in a temporary buffer associated with a pipeline process downstream of the current instruction. Figs. 3 illustrates an out-of-order processing system and Fig. 4 shows the interaction between a reservation station and a reorder buffer. Likewise, these figures do not show for detection of a dependency, determining for each current instruction involved in a renaming process that a logic target address of one or more instructions is not the same as a logic source address of said current instruction, said one or more instructions being stored in a temporary buffer associated with a pipeline process downstream of the current instruction. Column 8, lines 6-54 of Gaertner discuss assigning and the subsequent processing of physical registers to logical registers. It is noted that this discussion never mentions for detection of a dependency, determining for each current instruction involved in a renaming process that a logic target address of one or more instructions is not the same as a logic source address of said current instruction, said one or more instructions being stored in a temporary buffer associated with a pipeline process downstream of the current instruction.

For at least these reasons, claim 1 is believed allowable and indication of such allowance is earnestly requested.

<u>Claim 2:</u>

Claim 2 is amended to recite, in part, "in case <u>the logic target register addresses and the logic source register address match, setting the</u> no-dependency signal ~~is~~ <u>to</u> not active." Support for this claim limitation can be found at least at Fig. 7 and paragraph 66 of the present application.

Claim 2, as amended, overcomes the 35 USC §102 rejection since Gaertner does not disclose setting a no-dependency signal to not active when the logic target register addresses and the logic source register address match.

Specifically, nowhere in Gaertner is there is a discussion of no-dependency signal set to not active if a logic target register addresses and a logic source register address match.

For at least this reason, and the reasons given for claim 1, claim 2 is believed allowable and indication of such allowance is earnestly requested.

Claim 3:

Claim 3 recites, "The method according to claim 1 further comprising the step of evaluating 'valid'-bits of speculative target registers stored in a storage associated with speculatively calculated instruction result data to generate the no-dependency signal."

In rejecting claim 3, the Office Action states that "Gaertner has taught evaluating 'valid'-bits of speculative target registers stored in a storage associated with speculative calculated instruction result data to generate the no-dependency signal. FOA, ¶ 8, citing Gaertner, col. 8, ln. 6-54 and Figs. 2-4. The Applicant respectfully disagrees with such an interpretation of Gaertner.

Gaertner is directed to allocating and renaming registers in a computing system that processes instructions out-of-order. Gaertner, col. 1, ln. 6-10. Fig. 2 shows a program example executed out-of-order. It does not illustrate evaluating 'valid'-bits of speculative target registers stored in a storage associated with speculative calculated instruction result data to generate the no-dependency signal. Figs. 3 illustrates an out-of-order processing system and Fig. 4 shows the interaction between a reservation station and a reorder buffer. Likewise, these figures do not show evaluating 'valid'-bits of speculative target registers stored in a storage associated with speculative calculated instruction result data to generate the no-dependency signal. Column 8, lines 6-54 of Gaertner discuss assigning and the subsequent processing of physical registers to logical registers. It is noted that this discussion never mentions evaluating 'valid'-bits of speculative target registers stored in a storage associated with speculative calculated instruction result data to generate the no-dependency signal.

Thus, it is respectfully submitted that Gaertner does not describe the limitations of claim 3. For at least this reason, and the reasons given for claim 1, claim 3 is believed allowable and indication of such allowance is earnestly requested.

Claim 4:

Claim 4 recites, "addressing a mapping-table-entry with a logical source register address of said current instruction thus determining the mapped physical target register address."

In rejecting claim 4, the Office Action states that Gaertner teaches addressing a mapping-table-entry with a logical source register address of said current instruction thus determining the mapped physical target register address. FOA, ¶ 9, citing Gaertner, col. 8, ln. 6-54 and Figs. 2-4. The Applicant respectfully disagrees with such an interpretation of Gaertner.

Gaertner is directed to allocating and renaming registers in a computing system that processes instructions out-of-order. Gaertner, col. 1, ln. 6-10. Fig. 2 shows a program example executed out-of-order. It does not illustrate addressing a mapping-table-entry with a logical source register address of said current instruction thus determining the mapped physical target register address. Figs. 3 illustrates an out-of-order processing system and Fig. 4 shows the interaction between a reservation station and a reorder buffer. Likewise, these figures do not show addressing a mapping-table-entry with a logical source register address of said current instruction thus determining the mapped physical target register address. Column 8, lines 6-54 of Gaertner discuss assigning and the subsequent processing of physical registers to logical registers. It is noted that this discussion never mentions addressing a mapping-table-entry with a logical source register address of said current instruction thus determining the mapped physical target register address.

Claim 4 further recites, "reading a committed-status flag in said entry" and "generating a dependency-signal for the respective source register." Office Action states that Gaertner teaches reading a committed-status flag in said entry and generating a dependency-signal for the respective source register. FOA, ¶ 9, citing Gaertner, col. 9, ln. 66 - col. 10, ln. 32 and Fig. 4. The Applicant respectfully disagrees with such an interpretation of Gaertner.

Fig. 4 of Gaertner illustrates how the reservation station interacts with the reorder buffer, in order to correctly monitor the completion status, and to allow for precise interrupts. There is no teaching in this figure of either reading a committed-status flag in said entry or generating a dependency-signal for the respective source register. Furthermore, column 9, line 66 to column 10, line 32 similarly discuss the interplay between reservation station and the reorder buffer. The Applicant respectfully submits that there is no discussion of either reading a committed-status flag in said entry or generating a dependency-signal for the respective source register.

Thus, it is respectfully submitted that Gaertner does not describe the limitations of claim 4. For at least this reason, and the reasons given for claim 1, claim 4 is believed allowable and indication of such allowance is earnestly requested.

Claim 5:

Claim 5 is amended to recite, in part, "in case the logic target register address and the logic source register address match, setting the no-dependency signal ~~is~~ to not active." Support for this claim limitation can be found at least at Fig. 7 and paragraph 66 of the present application.

Claim 5, as amended, overcomes the 35 USC §102 rejection since Gaertner does not disclose setting a no-dependency signal to not active when the logic target register address and the logic source register address match. Specifically, nowhere in Gaertner is there is a discussion of no-dependency signal set to not active if a logic target register address and a logic source register address match.

For at least this reason, and the reasons given for claim 1, claim 5 is believed allowable and indication of such allowance is earnestly requested.

Claim 6:

Claim 6 is amended herein to recite, "a third computer readable code for bypassing a portion of the instruction pipeline for the current instruction if the no-dependency signal is active." Support for this claim limitation can be found at least at Fig. 6 and paragraph 62 of the present application. Claim 6, as amended, overcomes the 35 USC §102 rejection since Gaertner does not disclose bypassing a portion of the instruction pipeline for the current instruction if the no-dependency signal is active.

Gaertner is directed to allocating and renaming registers in a computing system that processes instructions out-of-order. Gaertner, col. 1, ln. 6-10. Specifically, an allocator is disclosed which is responsible for correctly addressing physical registers at the register renaming and allocation stage. Gaertner, col. 13, ln. 4-6 and Fig. 7. It is respectfully submitted that there is no teaching in Gaertner of bypassing a portion of an instruction pipeline for a current instruction if a no-dependency signal is active.

Claim 6 further recites, "a first computer readable code for, the detection of a dependency, determining for each current instruction involved in a renaming process that a logic target address of one or more instructions stored in a temporary buffer associated with a pipeline process downstream of the current instruction is not the same as a logic source address of said current instruction." In rejecting claim 6, the Office Action states that Gaertner teaches a first computer readable code for, the detection of a dependency, determining for each current instruction involved in a renaming process that a logic target address of one or more instructions stored in a temporary buffer associated with a pipeline process downstream of the current instruction is not the same as a logic source address of said current instruction. FOA, ¶ 10, citing Gaertner, col. 8, ln. 6-54 and Figs. 2-4. The Applicant respectfully disagrees with such an interpretation of Gaertner.

Fig. 2 shows a program example executed out-of-order. It does not illustrate a first computer readable code for, the detection of a dependency, determining for each current instruction involved in a renaming process that a logic target address of one or more instructions stored in a temporary buffer associated with a pipeline process downstream of the current instruction is not the same as a logic source address of said current instruction. Figs. 3 illustrates an out-of-order processing system and Fig. 4 shows the interaction between a reservation station and a reorder buffer. Likewise, these figures do not show a first computer readable code for, the detection of a dependency, determining for each current instruction involved in a renaming process that a logic target address of one or more instructions stored in a temporary buffer associated with a pipeline process downstream of the current instruction is not the same as a logic source address of said current instruction. Column 8, lines 6-54 of Gaertner discuss assigning and the subsequent processing of physical registers to logical registers. It is noted that this discussion never mentions a first computer readable code for, the detection of a dependency, determining for each current instruction involved in a renaming process that a logic target address of one or more instructions stored in a temporary buffer associated with a pipeline process downstream of the current instruction is not the same as a logic source address of said current instruction.

For at least these reasons, claim 6 is believed allowable and indication of such allowance is earnestly requested.

Claim 10:

Claim 10 recites, "a third computer readable code for assigning an entry in the temporary buffer to the logic source address of said current instruction if the no-dependency signal is not active; and a fourth computer readable code for issuing the instruction operand data to an instruction execution unit without assigning the entry in the temporary buffer to the logic source address of said current instruction if the no-dependency signal is active." In rejecting claim 10, the Office Action states that Gaertner teaches computer readable code for assigning an entry in the temporary buffer to the logic source address of said current instruction if the no-dependency signal is not active and for issuing the instruction operand data to an instruction execution unit without assigning the entry in the temporary buffer to the logic source address of said current instruction if the no-dependency signal is active. FOA, ¶ 10, citing Gaertner, col. 8, ln. 6-54 and Figs. 2-4. The Applicant respectfully disagrees with such an interpretation of Gaertner.

Fig. 2 shows a program example executed out-of-order. It does not illustrate computer readable code for assigning an entry in the temporary buffer to the logic source address of said current instruction if the no-dependency signal is not active and for issuing the instruction operand data to an instruction execution unit without assigning the entry in the temporary buffer to the logic source address of said current instruction if the no-dependency signal is active. Figs. 3 illustrates an out-of-order processing system and Fig. 4 shows the interaction between a reservation station and a reorder buffer. Likewise, these figures do not show computer readable code for assigning an entry in the temporary buffer to the logic source address of said current instruction if the no-dependency signal is not active and for issuing the instruction operand data to an instruction execution unit without assigning the entry in the temporary buffer to the logic source address of said current instruction if the no-dependency signal is active. Column 8, lines 6-54 of Gaertner discuss assigning and the subsequent processing of physical registers to logical registers. It is noted that this discussion never mentions computer readable code for assigning an entry in the temporary buffer to the logic source address of said current instruction if the no-dependency signal is not active and for issuing the instruction operand data to an instruction execution unit without assigning the entry in the temporary buffer to the logic source address of said current instruction if the no-dependency signal is active.

Claim 10 further recites, "a first computer readable code for, the detection of a dependency, determining for each current instruction involved in a renaming process that a logic target address of one or more instructions stored in a temporary buffer associated with a pipeline process downstream of the current instruction is not the same as a logic source address of said current instruction" and "a second computer readable code for generating a no-dependency signal associated with said current instruction." In rejecting claim 10, the Office Action states that Gaertner teaches computer readable code for the detection of a dependency, determining for each current instruction involved in a renaming process that a logic target address of one or more instructions stored in a temporary buffer associated with a pipeline process downstream of the current instruction is not the same as a logic source address of said current instruction and for generating a no-dependency signal associated with said current instruction. FOA, ¶ 10, citing Gaertner, col. 8, ln. 6-54 and Figs. 2-4. The Applicant respectfully disagrees with such an interpretation of Gaertner.

Fig. 2 shows a program example executed out-of-order. It does not illustrate computer readable code detection of a dependency, determining for each current instruction involved in a renaming process that a logic target address of one or more instructions stored in a temporary buffer associated with a pipeline process downstream of the current instruction is not the same as a logic source address of said current instruction and for generating a no-dependency signal associated with said current instruction. Figs. 3 illustrates an out-of-order processing system and Fig. 4 shows the interaction between a reservation station and a reorder buffer. Likewise, these figures do not show computer readable code for detection of a dependency, determining for each current instruction involved in a renaming process that a logic target address of one or more instructions stored in a temporary buffer associated with a pipeline process downstream of the current instruction is not the same as a logic source address of said current instruction and for generating a no-dependency signal associated with said current instruction. Column 8, lines 6-54 of Gaertner discuss assigning and the subsequent processing of physical registers to logical registers. It is noted that this discussion never mentions computer readable code for detection of a dependency, determining for each current instruction involved in a renaming process that a logic target address of one or more instructions stored in a temporary buffer associated with a pipeline process downstream of the current instruction is not the same as a logic source address of said current instruction and for generating a no-dependency signal associated with said current instruction.

CLAIM REJECTIONS UNDER 35 USC §103:

Claims 7-9 were rejected under 35 USC §103 as being obvious over Gaertner in view of U.S. Patent No. 5,974,526 to Garg et al. ("Garg").

A *prima facie* case for obviousness can only be made if the combined reference documents teach or suggest all the claim limitations. MPEP 2143.

Claim 7:

Claim 7 recites, "The processing system according to claim 6 in which in case of a content-addressable memory (CAM)-based renaming scheme the first computer readable code for determining the dependency of a current instruction comprises a compare logic in which all instructions to be checked for dependency are involved and an OR gate coupled with the compare logic." In rejecting claim 7, the Examiner states Garg teaches a content-addressable memory (CAM)-based renaming scheme. FOA, ¶ 13, citing Garg, col. 8, ln. 38-65 and col. 12, ln. 42-58. The Applicant respectfully disagrees with such an interpretation of Gaertner.

Column 8, lines 38-65 of Garg states,

> In one embodiment of the present invention, instructions can have from 0 to 3 inputs and 0 or 1 outputs. Most instructions' inputs and outputs come from, or are stored in, one of several register files. Each register file 117 (e.g., separate integer, floating and boolean register files) has 32 real entries plus the group of 8 temporary buffers 116. When an instruction completes, (The term "complete" means that the operation is complete and the operand is ready to be written to its destination register.) its result is stored in its preassigned location in the temporary buffers 116. Its result is later moved to the appropriate place in register file 117 after all previous instructions' results have been moved to their places in the register file. This movement of results from temporary buffers 116 to register file 117 is called "retirement" and is controlled by termination logic, as should become evident to those skilled in the art. More than one instruction may be retired at a time. Retirement comprises updating the "official state" of the machine, including the computer's Program. Counter, as will become evident to those skilled in the art. For example, if instruction I0 happens to complete directly before instruction I1, both results can be stored directly into register file 117. But if instruction I3 then completes, its result must be stored in temporary

> buffer 116 until instruction I2 completes. By having IEU 100 store each instruction's result in its preassigned place in the temporary buffers 116, IEU 100 can execute instructions out of program order and still avoid the problems caused by output and anti-dependencies. Garg, col. 8, ln. 38-65.

The Applicant submits there is no discussion of a content-addressable memory (CAM)-based renaming scheme mentioned in this citation of Garg. Furthermore, column 12, lines 42-58 of Garg states,

> In a preferred embodiment of the present invention, the inputs can come from the actual register file or an array temporary buffers 116. RRC 112 assumes that if an instruction has no dependencies, its inputs are all in the register file. In this case, RRC 112 passes the IXS1, IXS2 and IXS/D addresses that came from IFIFO 102 to the register file. If an instruction has a dependency, then RRC 112 assumes that the data is in temporary buffers 116. Since RRC 112 knows which previous instruction each instruction depends on, and since each instruction always writes to the same place in temporary buffers 116, RRC 112 can determine where in temporary buffers 116 an instruction's inputs are stored. It sends these addresses to register file read ports 119 and register file 117 outputs the data from temporary buffers 116 so that the instruction can use it. Garg, col. 12, ln. 42-58.

Again, the Applicant submits there is no discussion of a content-addressable memory (CAM)-based renaming scheme mentioned in this citation of Garg.

In relying upon the theory of inherency, the examiner must provide a basis in fact and/or technical reasoning to reasonably support the determination that the allegedly inherent characteristic necessarily flows from the teachings of the applied prior art. MPEP 2112 citing *Ex parte Levy,* 17 USPQ2d 1461, 1464 (Bd. Pat. App. & Inter. 1990) (emphasis in original).

The Exanimer states that the mapping-table based renaming system used by Garg is also inherently content-addressable, as it is indexed into using contents of its registers. FOA, ¶ 13. The Applicant respectfully submits that register indexing is not the same as content-addressable memory. While register indexing uses a register to provide a memory offest, content-addressable memory is designed such that the user supplies a data word and the CAM searches its entire memory to see if that data word is stored anywhere in it. If the data word is found, the CAM returns a list of one or more storage addresses where the word was found.

The Examiner indicates that item 206 of Fig. 2 and item 700 of Fig. 7 of Garg illustrate a compare logic in which all instructions to be checked for dependency are involved and an OR gate coupled with the compare logic. The Applicant respectfully submits that no OR gate coupled with the compare logic is shown in Figs. 2 and 7 of Garg.

For at least these reasons, and the reasons given for claim 6, claim 7 is believed allowable and indication of such allowance is earnestly requested.

Claim 8:

Claim 8 recites, "The processing system according to claim 7 further comprising a plurality of AND gates the input of which comprises a target register 'valid bits' signal and a respective compare logic output signal." In rejecting claim 8, the Examiner states Garg teaches a content-addressable memory (CAM)-based renaming scheme. FOA, ¶ 13, citing Garg, col. 8, ln. 38-65 and col. 12, ln. 42-58. The Applicant respectfully disagrees with such an interpretation of Gaertner.

The Examiner indicates that item 808 of Fig. 8 of Garg illustrate a plurality of AND gates the input of which comprises a target register 'valid bits' signal and a respective compare logic output signal. The Applicant respectfully submits that input of the AND gate shown in Fig. 8 of Garg. is coupled to the address of the source/destination of an instruction X (IXS/D) and the address of the source/destination of an instruction Y (IYS/D). Garg, col. 10, ln. 62 - col. 11, ln. 8. Thus, Garg does not teach or suggest a plurality of AND gates the input of which comprises a target register 'valid bits' signal and a respective compare logic output signal.

For at least this reason, and the reasons given for claims 6 and 7, claim 8 is believed allowable and indication of such allowance is earnestly requested.

Claim 9:

Claim 9 recites, "The processing system according to claim 6 in which the case of a mapping-table-based renaming scheme each mapping table entry comprises an additional instruction-commited flag, and the first computer readable code for determining the dependency of a current instruction comprises a logic for ANDing a target register 'valid bits' signal in which all instructions to be checked for dependency are involved and an OR gate coupled with the logic." In rejecting claim 8, the Examiner states Garg teaches a content-addressable memory (CAM)-based renaming scheme. FOA, ¶ 13, citing Garg, col. 8, ln. 38-65 and col. 12, ln. 42-58. The Applicant respectfully disagrees with such an interpretation of Gaertner.

The Examiner indicates that item 808 of Fig. 8 of Garg illustrate a plurality of AND gates the input of which comprises a target register 'valid bits' signal and a respective compare logic output signal. The Applicant respectfully submits that input of the AND gate shown in Fig. 8 of Garg. is coupled to the address of the source/destination of an instruction X (IXS/D) and the address of the source/destination of an instruction Y (IYS/D). Garg, col. 10, ln. 62 - col. 11, ln. 8. Thus, Garg does not teach or suggest a plurality of AND gates the input of which comprises a target register 'valid bits' signal and a respective compare logic output signal.

For at least this reason, and the reasons given for claims 6 and 7, claim 8 is believed allowable and indication of such allowance is earnestly requested.

CONCLUSION

In view of the forgoing remarks, it is respectfully submitted that this case is now in condition for allowance and such action is respectfully requested. If any points remain at issue that the Examiner feels could best be resolved by a telephone interview, the Examiner is urged to contact the attorney below.

No fee is believed due with this Amendment, however, should a fee be required please charge Deposit Account 50-0510. Should any extensions of time be required, please consider this a petition thereof and charge Deposit Account 50-0510 the required fee.

Respectfully submitted,

[illegible]

Dated: September 10, 2005

Ido Tuchman, Reg. No. 45,924
Law Office of Ido Tuchman
69-60 108th Street, Suite 503
Forest Hills, NY 11375
Telephone (718) 544-1110
Facsimile (718) 544-8588